Research Statement

SHAR Lwin Khin
School of Computing and Information Systems, Singapore Management University
Tel: (65) 6828-0019; Email: lkshar@smu.edu.sg
20 December 2024

Background

With the rise of Industry 4.0, software systems are becoming more pervasive in all industry sectors and have become increasingly complex and critical. As a result, the issues and challenges traditionally faced in software development are becoming more acute. They need more effective, efficient development processes, more automation, and more scalable solutions. Therefore, software engineering research is highly relevant and could have a significant impact to many industry sectors. However, there is a gap between academic-research and industrial needs in software engineering. In our software engineering research community, we tend to measure success by counting publications, which naturally lead to academics producing papers that may or may not be industry-relevant. A prominent software engineering researcher stated that among the publications from the top software engineering research venues, only a small proportion of the papers stem from industry-relevant research. My research aims to be industry-relevant. For this to happen, it must be context-driven.

Context-driven research doesn't try to frame a general problem and devise universal solutions. Rather, for a given context that is precisely defined, it makes clear working assumptions and devises practical solutions that work in such a context. It also considers and makes tradeoffs that make sense in the given context to achieve practicality and scalability. Clearly, this doesn't produce solutions that generalize easily to any arbitrary software development environment. But this doesn't reduce its value because universal solutions hardly work, anyway, for different software systems that have different contextual factors such as software complexity and characteristics, domain-related criticality and compliance with standards, organization's cost and time constraints, and human factors (e.g. engineers' background). Software from telecommunication industry could be very much different from those of automotive industry.

Whether the cost of a comprehensive software validation technique is justified will depend on the criticality of the software being assessed and the standards it must comply with. For example, in Singapore, software systems that acquire user privacy data must comply with Personal Data Protection Act (PDPA); hence a validation technique that specifically focuses on the PDPA will be beneficial for such systems. That is, they require a technique that can transform the natural language form of PDPA into a machine-analyzable form, that can extract the security and privacy requirements embedded in the PDPA, and that can support automated or semi-automated generation of test scenarios to validate these requirements. Although this may sound universal, in the nutshell, an effective technique requires domain-specific considerations — specific terms and definitions used in the PDPA must be considered for the natural language processing; testing typically requires specific set of inputs such as the inputs' form and content; and any other contextual factors such as the development process used, the cost and time constraints, etc. must be considered.

Research Areas

My research mainly focuses on security analysis of web/mobile/IoT applications and Cyber-Physical systems, which falls under the umbrella of software validation and verification domain – a sub-discipline of software engineering. Particularly, I work on detecting and analyzing *security vulnerabilities*, *privacy issues*, and *anomalies*. The kinds of software artefact applicable to my research range from source code to deployed software. My research is not limited to the use of any particular technique but for the problems that I am addressing, I found that the combined use of static analysis, dynamic analysis, search-based test generation, and machine learning techniques suits the need so far. The following discusses some of the major work that achieved high impact in my research domain:

Security Vulnerability Analysis [1-9]. The work proposed in ICSE NIER track [1] is one of the pioneer works that uses machine learning on code characteristics to predict vulnerabilities in web applications, differently from the then-approaches that mainly focused on the use of static and dynamic analyses only for detecting web vulnerabilities. In [1] and its subsequent extensions [2-4], we incorporated the use of machine learning to address the scalability and effectiveness problems of static and dynamic analyses; we proposed approaches that predict various code injection vulnerabilities (SQL injection, cross site scripting, remote code execution, path traversal, etc.) in web programs. Static and dynamic analyses are first used to extract code features from web programs containing known vulnerabilities. The extracted features reflect information about potentially correct and incorrect validation and sanitization routines implemented in a program. Supervised, semi-supervised, unsupervised machine learning techniques are then applied to build vulnerability predictors based on these features and the known, available vulnerability information. The predictors are then applied to predict vulnerabilities in other web programs. The experimental results show that the best predictor could detect 90% of vulnerabilities with 12% false alarm rate.

Vulnerability predictors have been shown to be effective, but they only predict vulnerable code sections. They do not provide comprehensive information on the deficiencies of the implementations that cause vulnerabilities. Therefore, to complement the above prediction approaches and to assist the developers in auditing the code, in [5, 6], I have also worked on a static analysis-based approach that specifically extracts security-relevant code slices for security auditing purposes. It also provides guidelines to carry out security audits based on the extracted code slices. This work addressed the technical and scalability challenges of precisely extracting the code slices for Java-based web programs.

In [8, 9], we extended this security auditing research in the area of improving the constraint solvers. This also augments existing vulnerability detection approaches. Given code slices, a powerful constraint solver will be able to tell precisely what inputs can cause the vulnerabilities. We developed such a constraint solver addressing the challenges of solving complex string operations and mixed (integer and string) operations in a scalable and effective way. For scalability, we built several finite-state machines (FSMs) that model various Java string operations, which avoid the need for resolving such operations into basic set of operations that are often recursive and

complex to solve; and then incorporated a search-based input generation algorithm into the FSMs-based constraint solver for effectiveness. The tool that implements the approach was evaluated on an industrial code base provided by HITEC Luxembourg (www.hitec.lu).

Privacy Analysis [10-13]. Even though security and privacy are often mentioned together, and many approaches claimed to address both issues at the same time, there are some subtle differences that pose specific challenges for addressing privacy concerns properly. Security has commonly accepted definitions and properties; for example, a secure communication system can be defined as the one that has confidentiality, integrity, and availability properties. By contrast, privacy is hard to be defined appropriately as it could mean differently to different people. To address this problem, in the scope of Android apps, in [10], we proposed an approach that reports anomalous data flows to users for them to make informed decisions. Firstly, using natural language processing and clustering techniques, our approach forms clusters of apps where each cluster contains a group of similar trusted, benign apps according to their functional descriptions; the approach then learns their normal behaviors by analyzing the call graphs of the app code and by running diverse test cases. We combined static analysis and genetic algorithm-based test generation so as to observe diverse behaviors as many as possible. Lastly, give a test app, the approach compares its behaviors against those of the apps in the same cluster. In [11], we extend the approach in [10] to detect permission re-delegation vulnerabilities that could leak privacy data. In [12], we proposed an approach that maps user interfaces in Android apps to permission uses so that users can make an informed privacy decisions when they interact with the apps. Even when privacy requirements are well-defined by stakeholders for privacy-sensitive systems, it is often the case that the requirements are only visible in the requirement documents; there is no traceability between the privacy specifications and the actual implementations. Hence, in this context, I have worked on a European industrial & academic research project called EDLAH2 (http://edlah2.eu), which attempts to address this problem. In this work, we developed a modeling method for specifying the security and privacy requirements in a traceable way [13].

Anomaly Analysis [14-20]. Concurrently with vulnerability and privacy analysis, I also conduct research in anomaly analysis that includes malware and generic software bugs. In the earlier work [14], I worked on malware behavior modeling using bounded feature space to deal with the scalability issue of signature-based malware detection approaches. Given that there is plethora of machine learning-based malware prediction approaches nowadays, I also worked on empirical studies to compare the malware detection performance of various types of features, classifiers (conventional classifiers and deep learning classifiers) [15, 21] and data preprocessing techniques [16].

I have also conducted research on anomaly detection in IoT and drone systems, in collaboration with HTX Singapore (https://www.htx.gov.sg/). Differently from traditional computing platforms, IoT computing platforms typically consist of diverse apps and devices ranging from sensors and cameras to smart cars and drones, and diverse communication protocols. And entities in IoT computing platform are highly automated, interactive, and inter-dependent. IoT devices may also be remote and have limited computing resources. Due to immaturity, IoT computing platforms are

also progressive. These unique characteristics make security analysis very challenging. Current approaches and tools are not yet adequate to these challenges [22]. In [17], we have developed an automated fuzzing approach for detecting anomalies in Samsung SmartThings IoT platform, which takes into consideration the interplays between SmartThings apps, devices, and user inputs. In [18-19], we proposed a machine learning-based analysis approach that detects anomalous drone behaviors in flight log data. The approach aims to detect anomalies such as sensor fault, actuator fault, configuration errors and bugs in drone control program. During a flight mission, drones typically log flight states, which could reflect anomalies. Hence, we train a LSTM-based deep learning model on the normal flight logs, which can then be used to detect anomalies in real time flight logs. In [20], we proposed an automated approach for configuring unsupervised learning systems in the context of detecting anomalies in Cyber-Physical systems.

In software engineering and empirical studies, it is important that tools that implements the proposed approaches and other software artefacts are made available to researchers for replication and extension. We have implemented several tools and made them available at

- Security auditing tool for web apps: https://github.com/julianthome
- Mobile apps security analysis tool: https://biniamf.github.io/PREV/
- Fuzzing tool for SmarThings apps: https://github.com/sharlwinkhin
- Anomaly detection tool for drones and empirical studies: https://github.com/Jesper20
- Fuzzing tool for drones: https://github.com/weiminn

Future Plan

Going forward, my long term plan is to improve the practice of software engineering through a context-driven research and produce a meaningful impact on industrial practice. I aim to realize this vision by conducting context-driven research in security testing & analysis in general with industrial collaborations, developing effective, efficient, (semi-)automated, and scalable solutions for testing their complex software systems under precise contexts.

As a short term plan, for the next two years, I plan to conduct translational research in the context of anomaly analysis of Cyber-Physical systems and vulnerability analysis of web applications through translational grants with industrial collaborators.

Selected Publications and Outputs

[1] **L.K. Shar** and H.B.K. Tan. Mining input sanitization patterns for predicting SQL injection and cross site scripting vulnerabilities. ICSE 2012.

[2] **L.K. Shar** et al. Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis. ICSE 2013.

[3] **L.K. Shar** et al. Web application vulnerability prediction using hybrid program analysis and machine learning. TDSC 2015.

[4] **L.K. Shar** and H.B.K. Tan. Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns. IST 2013.

[5] J. Thome, **L.K. Shar**, and L. Briand. Security Slicing for Auditing XML, XPath, and SQL Injection Vulnerabilities. ISSRE 2015.

[6] J. Thome, **L.K. Shar**, D. Bianculli, and L. Briand. Security slicing for auditing common injection vulnerabilities. JSS 2018.

[7] J. Thome, **L.K. Shar**, D. Bianculli, L. Briand. JoanAudit: A Tool for Auditing Common Injection Vulnerabilities. ESEC/FSE 2018.

[8] J. Thome, **L. K. Shar**, D. Bianculli, and L. Briand. Search-driven String Constraint Solving for Vulnerability Detection. ICSE 2017.

[9] J. Thome, **L.K. Shar**, D. Bianculli, L. Briand. An Integrated Approach for Effective Injection Vulnerability Analysis of Web Applications through Security Slicing and Hybrid Constraint Solving. TSE 2018.

[10] B.F. Demissie, M. Ceccato, and **L.K. Shar**. AnFlo: Detecting Anomalous Sensitive Information Flows in Android Apps. MobileSoft 2018.

[11] B.F. Demissie, B.F., M. Ceccato, **L.K. Shar**. Security analysis of permission re-delegation vulnerabilities in Android apps. EMSE 2020.

[12] V.K. Malviya, C.W. Leow, A. Kasthuri, Y.N. Tun, **L.K. Shar**, Right to Know, Right to Refuse: Towards UI Perception-Based Automated Fine-Grained Permission Controls for Android Apps. ASE 2022.

[13] X. P. Mai, A. Goknil, **L.K. Shar**, F. Pastore, L. Briand, S. Shaame. Modeling Security and Privacy Requirements: a Use Case-Driven Approach. IST 2018.

[14] M. Chandramohan, H.B.K. Tan, L.C. Briand, **L.K. Shar** and B. M. Padmanabhuni. A scalable approach for malware detection through bounded feature space behavior modeling. ASE 2013.

[15] **L.K. Shar** et al. Experimental comparison of features and classifiers for Android malware detection. MobileSoft 2020.

[16] **L.K. Shar** et al. Empirical evaluation of minority oversampling techniques in the context of Android malware detection, APSEC 2021.

[17] **L.K. Shar** et al. SmartFuzz: An Automated Smart Fuzzing Approach for Testing SmartThings Apps, APSEC 2020.

[18] **L.K. Shar** et al., DronLomaly: Runtime Detection of Anomalous Drone Behaviors via Log Analysis and Deep Learning, APSEC 2022.

[19] W. Minn, Y.N. Tun, **L.K. Shar**, L. Jiang, DronLomaly: Runtime Log-based Anomaly Detector for DJI Drones, ICSE Demonstrations 2024.

[20] **L.K. Shar** et al. AutoConf: Automated Configuration of Unsupervised Learning Systems Using Metamorphic Testing and Bayesian Optimization, ASE 2023.

[21] **L.K. Shar** et al. Experimental comparison of features, analyses, and classifiers for Android malware detection. EMSE 2023.

[22] V.K Malviya, W. Minn, **L.K. Shar**, L Jiang. Fuzzing drones for anomaly detection: A systematic literature review. Computers & Security 2024.